日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　９月２５日

出願番号
Application Number: 特願２００１－２９０９８６

出願人
Applicant(s): 株式会社日立製作所

U.S. Appln. Filed 2-15-02
Inventor: S. Misaka et al
Mattingly Stanger & Malur
Docket NIT-321

２００１年１２月　７日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３１０７５５４

【書類名】　特許願

【整理番号】　NT01P0324

【提出日】　平成13年 9月25日

【あて先】　特許庁長官　殿

【国際特許分類】　G06F 9/30
G06F 12/02
G06F 9/42

【発明者】

【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

【氏名】　三坂　智

【発明者】

【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

【氏名】　相坂　一夫

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社日立製作所

【代理人】

【識別番号】　100068504

【弁理士】

【氏名又は名称】　小川　勝男

【電話番号】　03-3661-0071

【選任した代理人】

【識別番号】　100086656

【弁理士】

【氏名又は名称】　田中　恭助

【電話番号】　03-3661-0071

【選任した代理人】

【識別番号】　100094352

【弁理士】

【氏名又は名称】　佐々木　孝

【電話番号】　03-3661-0071

【手数料の表示】

【予納台帳番号】　081423

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　異種実行環境における計算機システム構成方法、および、異種実行環境で実行させるためのプログラム

【特許請求の範囲】

【請求項１】　計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際の異種実行環境における計算機システム構成方法において、

この計算機システムは、ハードウェアとして汎用レジスタを有するＣＰＵと、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

前記ＣＰＵの命令セットとしては、前記汎用レジスタにＩｍｍｅｄｉａｔｅ値を格納するＩｍｍｅｄｉａｔｅロード命令を含み、

前記オペレーティングシステム上で実行させる実行プログラムとしては、前記アプリケーション・プログラムに対応する実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムから前記アプリケーション・プログラムに対応する実行プログラムに、制御が戻るときに返すエラーコードを実行環境によらない共通化エラーコードとして定義しておき、

しかも、この共通化エラーコードを、前記Ｉｍｍｅｄｉａｔｅロード命令で設定可能な数値範囲内の値として定め、

前記共通化エラーコードを、前記Ｉｍｍｅｄｉａｔｅロード命令の命令コード内に保持することを特徴とする異種実行環境における計算機システム構成方法。

【請求項２】　前記共通化エラーコードを、前記Ｉｍｍｅｄｉａｔｅ値のＭＳＢをゼロとした数値範囲内で定めることを特徴とする請求項１記載の異種実行環境における計算機システム構成方法。

【請求項３】　前記命令セットの仕様として、前記汎用レジスタのビット数より小さいデータを、前記汎用レジスタにロードするときに、そのデータの先頭部を自動的にゼロ拡張するようになっているときに、

前記共通化エラーコードを、符号無しの正の数で定めることを特徴とする請求項１記載の異種実行環境における計算機システム構成方法。

【請求項４】　前記命令セットの仕様として、前記汎用レジスタのビット数より小さいデータを、前記汎用レジスタにロードするときに、そのデータの先頭部を自動的に符号拡張するようになっているときに、

前記共通化エラーコードを、前記Ｉｍｍｅｄｉａｔｅ値のＭＳＢをゼロとした数値範囲内で定めることを特徴とする請求項１記載の異種実行環境における計算機システム構成方法。

【請求項５】　計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際の異種実行環境で実行させるためのプログラムにおいて、

この計算機システムは、ハードウェアとして汎用レジスタを有するＣＰＵと、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

前記ＣＰＵの命令セットとしては、前記汎用レジスタにＩｍｍｅｄｉａｔｅ値を格納するＩｍｍｅｄｉａｔｅロード命令を含み、

前記オペレーティングシステム上で実行させる実行プログラムとしては、前記アプリケーション・プログラムに対応する実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムから前記アプリケーション・プログラムに対応する実行プログラムに、制御が戻るときに返すエラーコードを実行環境によらない共通化エラーコードとして定義しておき、

しかも、この共通化エラーコードを、前記Ｉｍｍｅｄｉａｔｅロード命令で設定可能な数値範囲内の値として定め、

前記共通化エラーコードを、前記Ｉｍｍｅｄｉａｔｅロード命令の命令コード内に保持して、

前記アプリケーション・プログラムに対応する実行プログラム内のコードと前記実行環境差異吸収プログラム内のコードにのみ共通化エラーコードの情報を含むことを特徴とする異種実行環境で実行させるためのプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、異種実行環境における計算機システム構成方法に係り、アプリケーション・プログラムに、異なった実行環境の共通のエラーコードを返すようにするときに、実行プログラムのメモリ効率がよく、実行速度を低下させることなく、計算機システムを実装できるような計算機システム構成方法に関する。

【０００２】

【従来の技術】

従来、異なった環境でコンピュータシステムを実行させるために、開発や移植を容易におこなうための技術が提案されている。

【０００３】

このような技術の例として、特開平８－６３３６３号公報の「仮想実行環境システム」がある。

【０００４】

以下では、図１を用いてこの仮想実行環境システムについて説明する。

図１は、特開平８－６３３６３号公報の「仮想実行環境システム」のシステム概要図である。

【０００５】

ユーザ（プログラマ）が作成するアプリケーション・ソフトウェアのソースコードとしては、プログラム本体１１と置換情報記述部１２がある。

【０００６】

プログラム本体は、アプリケーション・ソフトウェアの目的ごとに異なるソースコードであり、置換情報記述部１２は、プログラム本体１１中に記述された情報を、仮想実行環境が実現される既存オペレーティング・システム６０の機種毎に対応して適切な情報に置換える部分である。

【０００７】

オペレーティング・システム６０で実行するためには、これらのプログラム本体１１と置換情報記述部１２に記述された置換情報を、既存オペレーティング・システム用コンパイラを利用して、実行可能プログラム３０にし、それを実行部２２が実行する。翻訳部２１は、そのための仮想実行環境上で実行可能な実行可能プログラムに翻訳する部分である。

【０００８】

その際に、ソースプログラムをＣ言語で記述するときには、置換情報記述部１２は、コンパイラのプリプロセッサとして処理されるマクロ定義を利用することができる。

【０００９】

このようにして、アプリケーションソフトウェア１０のプログラム本体１１を変更する事無く、様々な環境用に作成された実行可能プログラム３０を既存オペレーティング・システム６０上で動作させることが可能となる。

【００１０】

次に、図２を用いて、より詳細に異種実行環境において従来技術の計算機システムを構成し、実行する場合について説明する。

図２は、異種実行環境においてソフトウェア開発をおこなうときの概念図である。

【００１１】

ここでは、Ｃ言語により、アプリケーションを開発する場合を想定する。図１の置換情報記述部１２にあたる部分に、Ｃ言語の共通関数１０２が提供される。

【００１２】

システムとして、図２の様に計算機システムーＡ１０５ａ、計算機システムーＢ１０５ｂがあり、そこでプログラムを実行する環境を実行環境－Ａ１０４ａ、実行環境－Ｂ１０４ｂと言うことにする。実行環境－Ａ１０４ａは、具体的には、ハードウェアーＡ１０６ａと、ハードウェアを制御しアプリケーションプログラムとハードウェアの仲立ちをするＯＳ－Ａ１０７ａである。

【００１３】

アプリケーション本体１００は、ユーザが個々のアプリケーションを記述するソースコードであり、システムから機能の提供を受けるときには、共通化関数１０２を呼出す。ここで、注意することは、アプリケーション本体１００を記述するユーザは、共通化関数インタフェース１０１を意識すればよく、個々の環境によってコーディングを変えなくても良いことである。したがって、アプリケーション本体１００は、ソースレベルでの互換性が保たれることになる。

【００１４】

共通化関数１０２、開発用ライブラリ関数として提供され、ユーザには、共通化関数インタフェースが公開されることになる。共通化関数は、この様に環境に依らない統一したインタフェースをユーザに提供するためのものである。

【００１５】

そして、ユーザが記述したソースをコンパイル・リンク作業をおこなって、実行形式を作成することになるが、実行環境－Ａ１０４ａで実行させるときには、図２に示される様に、コンパイラーＡ１０８ａでコンパイルし、実行環境－Ｂ１０４ｂで実行させるときには、コンパイラーＢ１０８ｂでコンパイルする。

【００１６】

また、同様に実行環境－Ａ用のリンカＡ１０９ａにより、実行環境プログラム－Ａ１０３ａをリンクする。実行環境プログラム－Ａ１０３ａは、実行環境－Ａ１０４ａで実行させるためのプログラムであり、このなかで、ＯＳ－Ａ１０７ａのシステムコールがコールされることになる。実行環境－Ｂ１０４ｂについても同様である。

【００１７】

ユーザのソースコードであるアプリケーション本体１００に対して、コンパイル、リンクがおこなわれ、この例では、実行環境－Ａ１０４ａと実行環境－Ｂ１０４ｂ用に、それぞれ実行プログラム－Ａ１１０ａと、実行プログラム－Ｂ１１０ｂが生成される。

【００１８】

そして、共通化関数１０２をコンパイルしたモジュールと、実行環境プログラム－Ａ１０３ａ、実行環境プログラム－Ｂ１０３ｂは、コンパイル・リンクの作業を経て、それぞれの固有の実行環境で実行可能な実行環境差異吸収プログラム－Ａ１１１ａと実行環境差異吸収プログラム－Ｂ１１１ｂになる。

【００１９】

次に、このようにして作成した実行プログラムを、計算機システムの実行環境のもとで動作させる場合について概説する。

【００２０】

例えば、実行プログラム－Ａ１１０ａを、実行環境－Ａのもとで実行するときには以下のようになる。すなわち、実行プログラム－Ａ１１０ａでは、共通化インターフェース１０２により、実行環境差異吸収プログラム－Ａ１１１ａを呼出す。この実行環境差異吸収プログラム－Ａ１１１ａは、アプリケーションから見たＯＳの差異を吸収し、インターフェースを共通化されるために設けられるモジュールである。呼び出された実行環境差異吸収プログラム－Ａ１１１ａは、ＯＳ－Ａ１０７ａの機能を実現するためのシステムコールを呼出して、所望のＯＳの機能をハードウェア－Ａ１０６ａにより実現する。ハードウェア－Ａ１０６ａの動作が終了すると、ＯＳ－Ａ１０７ａは、実行環境差異吸収プログラム－Ａ１１１ａに制御を戻す。最後に、実行環境差異吸収プログラム－Ａ１１１ａは、制御を実行プログラム－Ａ１１０ａに戻す。

【００２１】

実行プログラム－Ｂ１１０ｂを、実行環境－Ｂ１０４ｂのもとで実行させるときも、同様の動作をするが、実行環境－Ｂ１０４ｂは、計算機システム－Ｂでさせるときの環境であるので、前提となるＯＳやハードウェアが異なっている。例えば、ハードウェア－Ａ１０６ａとハードウェア－Ｂ１０６ｂでは、機械語命令体系が異なり、実行環境プログラム－Ａ１０３ａと実行環境プログラム－Ｂ１０３ｂでは、ＯＳの機能を呼出すためのシステムコールが異なっている。

【００２２】

共通化インタフェース１０２は、プログラム本体１１に変更を加える事無く、異なるＯＳ上に実装してもプログラム本体１１から、ほぼ同一のＯＳ機能を呼び出して、異なるＯＳ１０７に依存しない共通化エラーコードをプログラム本体１１に伝えることを可能にするものである。なお、ここで、共通化エラーコードと言っているのは、ＯＳに依存するエラーコードを、実行環境差異吸収プログラムで、そのまま、あるいは、書き換えて、アプリケーションから見たときには、環境によらない共通化されたエラーコードにするためである。

【００２３】

共通化インタフェース１０２の仕様を定義することでＯＳ機能の呼び出し方やプログラミング記述が統一され、プログラム本体１１を別のアプリケーションソ

フトウェアに再利用させることが容易になる。

【００２４】

【発明が解決しようとする課題】

上記従来技術は、異なった計算機システムの環境に、アプリケーションソフトウェアを開発して実行するシステムに関するものであった。

【００２５】

ここで、図３ないし図７を用いて上記従来技術の動作において、エラーコードの取扱いに関し問題点が生ずることを説明する。

図３は、計算機システムーＡ１０５ａで取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【００２６】

計算機システムーＡ１０５ａは、ハードウェアーＡ１０６ａと、ＯＳーＡ１０７ａと、実行環境差異吸収プログラムーＡ１１１ａと、実行プログラムーＡ１１０ａで構成されることは前述した。

【００２７】

ハードウェアーＡ１０６ａには、図３に示されるように命令を実行するＣＰＵーＡ１２１ａと、プログラムやデータを格納するＲＯＭーＡ１２２ａとＲＡＭーＡ１２３ａが搭載されている。そして、ＣＰＵーＡ１２１ａは、複数の汎用レジスタを有している。

【００２８】

すなわち、各用途により、ユーザの記述した引数を渡すための引数格納用途レジスタ１２４、一時変数を格納する一時変数格納用途レジスタ１２５、Ｃ関数の戻り値を返すための戻り値格納用途レジスタ１２６、ユーザに提供されるユーザ使用汎用レジスタ１２７などがある。

【００２９】

また、ＣＰＵーＡ１２１ａは、ＲＡＭーＡ１２３ａ上に実装されているスタック領域１３０内の番地を指すスタックポインタレジスタ１２８、ＲＯＭーＡ１２２ａまたはＲＡＭーＡ１２３ａ上に実装されたプログラムの現在実行中の命令の次のアドレスを指すプログラムカウンタレジスタ１２９を有している。

【００３０】

一般に、Ｃ言語などのコンパイラが、ＣＰＵの汎用レジスタをデータや引数に割り付けるときには、大別してユーザに対してある関数を呼出した前後の値の存在を保証して、ユーザが使用しても良いとする（すなわち、アセンブラでレジスタの値を直接操作をしても良い）レジスタと、システム上で利用する汎用レジスタがある。この汎用レジスタは、ユーザに対しては、ある関数を呼出した前後の値の存在は保証していない。

【００３１】

ここでは、任意のＣ関数が呼ばれた時点の前後で値の一貫性を保証しないレジスタとして、引数格納用途レジスタ１２４、一時変数格納用途レジスタ１２５、戻り値格納用途レジスタ１２６がある。また、一方、値の一貫性を保証するレジスタは、ユーザ使用汎用レジスタ１２７、スタックポインタレジスタ１２８、プログラムカウンタレジスタ１２９がある。

【００３２】

引数格納用途レジスタ１２４、一時変数格納用途レジスタ１２５、戻り値格納用途レジスタ１２６、ユーザ使用汎用レジスタ１２７、スタックポインタレジスタ１２８は、通常は、それぞれ、同じ大きさの値を扱える分のレジスタ長になっている。

【００３３】

ここで、ＲＯＭ－Ａ１２２ａ上に、実行プログラム－Ａ１１０ａのプログラムコードである実行プログラムコード－Ａ１３１ａと、実行環境差異吸収プログラム－Ａ１１１ａのプログラムコードである実行環境差異吸収プログラムコード－Ａ１３２ａと、ＯＳ－Ａ１０７ａのプログラムコードであるＯＳプログラムコード－Ａ１３３ａが格納されるものとする。これは、図２に示した様に、コンパイラＡ１０８ａにより、実行環境－Ａ用にコンパイルされたものである。そして、実行プログラム－Ａ１１０ａを実行させるためには、実行プログラムコード－Ａ１３１ａの命令語が解釈・実行される。また、同様に、実行環境差異吸収プログラム－Ａ１１１ａを実行させるためには、実行環境差異吸収プログラムコード－Ａ１３２ａ命令語が、ＯＳ－Ａ１０７ａのプログラムを実行するためには、Ｏ

Ｓプログラムコード－Ａ１３３ａが、それぞれ解釈・実行される。

【００３４】

また、ＲＯＭ－Ａ１２２ａ上は、定数格納領域－Ａ１３４ａが実装されており、ＯＳ－Ａ１０７ａの機能に依存したＯＳ依存エラーコード１３５ａと、実行環境差異吸収プログラムで共通化されて提供される共通化エラーコード１３６が格納されている。

【００３５】

さらに、ＲＡＭ－Ａ１２３ａ上には、既述のようにスタック領域１３０が実装されている。

【００３６】

実行プログラムコード－Ａ１３１ａを実行するときには、ＣＰＵ－Ａ１２１ａにより、実行プログラム開始番地１３７から始まり、実行プログラム終了番地１３８までにある命令語が解釈・実行される。その際、プログラムカウンタレジスタ１２９は、現在動作させている機械語命令の次に動作させる機械語命令の格納番地を指しており、スタックポインタレジスタ１２８は、現在使用しているスタック領域１３０内の任意のメモリ番地を指している。

【００３７】

プログラムカウンタレジスタ１２９は、実行プログラムコード－Ａ１３１ａ内にプログラムされた共通化インターフェース１０２部の入口部に到達すると、ジャンプ命令１３９によって、その共通化インターフェース１０２に対応する実行環境差異吸収プログラムコード－Ａ１３２ａの先頭番地を指してＣＰＵ－Ａ１２１ａを実行する。

【００３８】

同じく、プログラムカウンタレジスタ１２９は、実行環境差異吸収プログラムコード－Ａ１３２ａ内にプログラムされたＯＳ－Ａ１０７ａのシステムコールの入口部に到達すると、ソフトウェア割り込み命令またはジャンプ命令１４０により、そのシステムコールに対応するＯＳプログラムコード－Ａ１３３ａの先頭番地を指して、ＣＰＵ－Ａ１２１ａを実行する。

【００３９】

ＯＳプログラムコードーＡ１３３ａが所定の機能を実行し終えると、実行環境差異吸収プログラムーＡ１１１ａに伝えるため、ＯＳーＡ１０７ａ依存エラーコード１３５ａを、戻り値格納用途レジスタ１２６に戻り値を格納する。ＯＳーＡ１０７ａ依存エラーコード１３５ａは、ＯＳ１０７ａのプログラムが正常終了したのか、ＯＳ１０７ａのプログラムや計算機システムーＡ１０５ａ内部でどのような異常が発生したかを示すための情報である。

【００４０】

ＯＳーＡプログラムコードーＡ１３３ａによる戻り値の格納作業を終えると、プログラムカウンタレジスタ１２９は、実行環境差異吸収プログラムコードーＡ１３２ａへの戻り先番地１４２を指して、リターン命令１４１を実行することにより、実行環境差異吸収プログラムコードーＡ１３２ａに制御を移す。

【００４１】

実行環境差異吸収プログラムコードーＡ１３２ａに制御が戻ると、戻り値格納用途レジスタ１２６には、ＯＳーＡ１０７ａ依存エラーコード１３５ａが戻り値として入っている。一方、ＲＯＭーＡ１２２ａ内の或るＯＳーＡ１０７ａ依存エラーコード１３５ａを一時変数格納用途レジスタ１２５にロードして、そして比較命令１４４で戻り値格納用途レジスタ１２６と一時変数格納用途レジスタ１２５に格納されている値を比較する。そして、全く等価なエラーコードが見つかったときには、実行環境差異吸収プログラムコードーＡ１３２ａから、実行プログラムコードーＡ１３１ａに制御が戻るときに、その全く等価なエラーコードに対応する共通化エラーコード１３６を、戻り値格納用途レジスタ１２６に格納して、実行プログラムーＡ１１０ａに渡すことにする。見つからなかったときには、次の或るＯＳーＡ１０７ａ依存エラーコード１３５ａを一時変数格納用途レジスタ１２５にロードして、比較命令１４４を繰りかえす。ここで、言う等価なエラーコードを見つけるとは、例えば、ＯＳーＡ１０７ａ依存エラーコード１３５ａが、「メモリエラー」を意味するときには、同じ「メモリエラー」を意味する共通化エラーコードを見つけると言うことである。

【００４２】

ここで、ポイントになるのは、実行環境ーＡ１０４ａと実行環境ーＢ１０４ｂ

とは、エラーコードは、共通化されており、実行プログラム－Ａ１１０ａと実行プログラム－Ｂ１１０ｂには、同じエラーコードが渡されると言うことである。そのため、アプリケーションを記述する者にとっては、環境の差異により返されるエラーコードが変わると言うことがなくなり、環境の差異によるコーディングの変更を気にしなくても良いと言うメリットが生ずる。

【００４３】

なお、実行開発環境１０４によっては、ＯＳ依存エラーコード１３５と共通化エラーコード１３６は、それぞれ、定数格納領域１３４内ではなく、実行プログラムコード１３１、実行環境差異吸収プログラムコード１３２の途中等に実装されることもある。

【００４４】

上記従来技術では、実行環境の差異を吸収するレイヤを設けたため、ＯＳに依存しない共通化エラーコードを定義して、ユーザにアプリケーションの移植・記述の容易性を提供することができた。しかしながら、エラーコードのインプリメントに関しては、考慮されていない。

【００４５】

というのも、共通化エラーコード１３６は、ＲＯＭ－Ａ１２２ａの定数格納領域１３４、そして、実行プログラムコード１３１、および、実行環境差異吸収プログラムコード１３２の内部に格納されるために、ＲＯＭ効率が悪くなるためである。この問題点は、共通化エラーコード１３６を大きな格納領域を要する値で定義した場合には、より大きくなるためである。

【００４６】

このために、共通化エラーコード１３６の値を小さくすることが有効であるが、そのようにすると、ＣＰＵの命令セットの仕様から新たな問題点を生じる。以下、それについて、図４ないし図７を用いて説明する。

図４は、汎用レジスタでゼロ拡張する際の説明図である。

図５は、ゼロ拡張された汎用レジスタの値を負にする際の説明図である。

図６は、汎用レジスタで符号拡張する際の説明図である。

図７は、符号拡張された汎用レジスタの値をもとに戻す際の説明図である。

【００４７】

ＣＰＵの命令セットの仕様としては、汎用レジスタに格納可能な範囲より比較的小さな値を格納する際に先ずゼロ拡張するもの、同じく先ず符号拡張するものが現存する。

【００４８】

図４のように、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、ゼロ拡張されるものがある。このときには、上の例では、ＣＰＵ１２１αは、小さな正の値である定数データを、戻り値格納用途レジスタ１２６または一時変数格納用途レジスタ１２５の下位部１６１に格納し、上位部１６２をゼロ拡張する作業を１命令でおこなうことになる。

【００４９】

また、図５に示されように、ゼロ拡張された汎用レジスタの値を負にするためには、上位部に０を２の補数変換処理命令１６３を施すなどの処理が必要になる。

【００５０】

したがって、共通化エラーコード１３６の値を小さくしても、共通化エラーコード１３６の値を負のコードで定義して、しかも、レジスタにロードするときの命令セットの仕様が、上記のようにゼロ拡張するものであるときには、２の補数変換処理命令１６３等が追加することが必要になり、その分だけ、実行プログラムコードーＡ１３１ａや実行環境差異吸収プログラムコードーＡ１３２ａの命令数が増え、ＲＯＭ効率が悪くなると言う問題点が生ずる。

【００５１】

一方、図６のように、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、符号拡張されるものがある。このときには、上の例では、ＣＰＵ１２１βは、小さな正の値である定数データを、戻り値格納用途レジスタ１２６または一時変数格納用途レジスタ１２５の下位部１６１に格納する。そして、小さな値の符号ビットが０（正）のときには、レジスタの上位部を０にし、１（負）のときには、レジスタの上位部を１にする。この一連の動作は、一命令でおこなわれることになる。

【００５２】

図７は、負に符号拡張された汎用レジスタの値を、もとに戻す処理を示している。もし、小さな値の定数データのＭＳＢ（Most Significant Bit）が１であるとする。このときに、この定数データを符号無しデータとみなすか、符号付きデータでみなすかで表している数値の意味が異なってくる。

【００５３】

そして、小さな値の定数データが、ＭＳＢが１の符号無しデータであり、かつ、図６に示したように符号拡張されたときには、上位部の１を再び０にする必要が生じる。

【００５４】

このように、共通化エラーコード１３６の値を小さくして、共通化エラーコード１３６の値を符号無しのコードで定義しても、レジスタにロードするときの命令セットの仕様が、上記のように符号拡張するものであるときには、上位部に０をパックする命令１６５等が追加することが必要になり、その分だけ、実行プログラムコードーＡ１３１ａや実行環境差異吸収プログラムコードーＡ１３２ａの命令数が増え、ＲＯＭ効率が悪くなると言う問題点が生ずる。

【００５５】

本発明は、上記問題点を解決するためになされたもので、その目的は、計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際に、既存の命令セットの体系を活かしながら、その共通化コードのインプリメントのためのＲＯＭ効率を良くし、むだな命令を挿入することのない計算機システム構成方法を提供することにある。

【００５６】

【課題を解決するための手段】

本発明では、実行環境の差異を吸収するために定義されている共通化エラーコード１３６を、定数格納領域１３４に格納しなくても良いようにするため、実行環境差異吸収プログラムコード１３２からの戻り値と共通化エラーコード１３６の値を同じ値にし、ＣＰＵ１２１の実行できるＩｍｍｅｄｉａｔｅロード命令１８１に埋め込める範囲内とするという手段を採る。

【００５７】

また、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、ゼロ拡張されるようになっているとき、２の補数変換処理命令１６３等を追加しなくても良いようにするために、共通化エラーコード１３６を符号無しの正のデータとして定義する。

【００５８】

このように共通化エラーコード１３６を符号無しの正のデータと定義することで、２の補数変換処理命令１６３を追加しなくても良くなる。

【００５９】

したがって、実行プログラムコード１３１や実行環境差異吸収プログラムコード１３２において、そのような命令を追加しなくても良くなり、ＲＯＭ効率を良くすることができ、そのような命令を実行しなくて良いため、性能の低下も防止できる。

【００６０】

また、命令セットの仕様として、小さな値を汎用レジスタにロードするときには、符号拡張されるようになっているとき、ゼロ埋め込み命令１６５を追加しなくても良いようにするため、共通化エラーコード１３６の値を、Ｉｍｍｅｄｉａｔｅ設定部１８２のＭＳＢをゼロとして表せる数値範囲内にする。

【００６１】

このように共通化エラーコード１３６の値を、Ｉｍｍｅｄｉａｔｅ設定部１８２のＭＳＢをゼロとして表せる数値範囲内にすることで、ゼロ埋め込み命令１６５を追加しなくても良くなる。

【００６２】

したがって、共通化エラーコード１３６を、Ｉｍｍｅｄｉａｔｅ設定部１８２のＭＳＢをゼロとして表せる範囲の値に定義することで、ゼロ埋め込み命令１６５の追加は考慮せずに済むことが可能となる。

【００６３】

したがって、実行プログラムコード１３１や実行環境差異吸収プログラムコード１３２において、そのような命令を追加しなくても良くなり、ＲＯＭ効率を良

くすることができ、そのような命令を実行しなくて良いため、性能の低下も防止できる。

【００６４】

上記のように共通化エラーコード１３６を設定することで、命令セットとして、Ｉｍｍｅｄｉａｔｅロード命令を備えるＣＰＵ１２１に対し、実行プログラムコード１３１や実行環境差異吸収プログラムコード１３２に、機械語命令を追加したり、定数格納領域１３４の容量を増加させることなく、ＲＯＭ効率を向上させることが可能になる。

【００６５】

【発明の実施の形態】

以下、本発明に係る一実施形態を、図８ないし図１６を用いて説明する。

〔開発環境とハードウェア〕

先ず、図８および図９を用いて本発明に係る異種実行環境における計算機システム構成方法のソフトウェア開発環境と、それを動作させるためのハードウェアについて説明する。なお、本実施形態では、ソースプログラムの記述を、Ｃ言語でおこなった場合を想定するものとする。

図８は、本発明に係る異種実行環境における計算機システム構成方法のソフトウェア開発環境の処理の流れを説明するための図である。

図９は、本発明に係る異種実行環境における計算機システム構成方法を実行するためのハードウェア構成図である。

【００６６】

本発明は、複数の異なったＯＳでの開発環境２１０を前提としている。

【００６７】

ソフトウェアの開発者は、アプリケーションのプログラムが記述したアプリケーション本体１００と、共通化関数１０２のソースコードを入力して、通常の開発手順の通り、プリプロセッサ２１１、Ｃコンパイラ２１２、アセンブラ２１３、リンカ２１４と言うソフトウェアツールにより、実行プログラムを作成する。

【００６８】

プリプロセッサ２１１は、条件によって加工したり、読み飛ばしたり、他プロ

グラムを組み込んだりマクロ定義処理などをおこなうＣコンパイラに入る前の前処理のためのツールである。Ｃコンパイラ２１２は、記述されたテキストコードに対して、構文解析・意味解析などをおこない、適当なアセンブラコードを生成するものである。アセンブラ２１３は、アセンブラコードをＣＰＵ１２１で実行させる機械語に変換するツールである。

【００６９】

リンカ２１４は、ライブラリ１９０から適当なプログラムコードを選び、数種のプログラムコードファイルを結合して最終的にハードウェア上で実行させる実行プログラムを作成する。本実施形態の例では、アプリケーション本体１００をコンパイルした実行プログラムコード１３１、実行環境差異吸収プログラムコード１３２、および、ＯＳのコードといった必要なモジュールと、定数格納領域１３４、スタック領域１３０などのデータ部分とを結合させて、実際にＣＰＵ１２１に実行させる実行形式のモジュールを作成する。このライブラリ１９０には、実効環境に依存する実行環境プログラムコードが含まれていて、機能に応じてリンカ２１４により適当なプログラムが取りこまれて結合される。

【００７０】

ＯＳコンフィグレータ１９１は、アプリケーションソフトウェア開発者に、ターゲットとするＯＳ１０７の諸機能を選択および設定させカスタマイズ化されたＯＳ１０７のプログラムコードを作成するソフトウェアツールである。

【００７１】

ハードウェア１０６は、ＣＰＵ１２１、Ｄｅｖｉｃｅ２０１、ワーキングメモリ２０２から構成されている。

【００７２】

最終的にＣＰＵ１２１上で実行する実行プログラムとＯＳ１０７は、ワーキングメモリ２０２に配置される。

【００７３】

ワーキングメモリ２０２は、ＲＯＭとＲＡＭであり、実行形式のモジュールとして、ユーザの実行プログラムと実行環境差異吸収プログラム、ＯＳのモジュールが格納される。また、データとしては、定数格納領域１３４、スタック領域１

３０が格納される。

【００７４】

本実施形態のソフトウェア開発環境は、メインフレーム、パーソナルコンピュータやワークステーション等の汎用計算機上に実装することが可能である。

【００７５】

最終的に実行プログラムを動作させることのできるハードウェア１０６を作成する手順を示すと以下の様になる。

【００７６】

すなわち、アプリケーションソフトウェア開発者は、汎用計算機上に定義されているターゲットのＯＳ１０７に対応するＯＳ別プログラム開発環境２１０を選択する。そして、選択されたＯＳ別プログラム開発環境２１０に付随しているＯＳコンフィグレータ１９１を用いてカスタマイズしたＯＳ１０７のプログラムコードを作成する。

【００７７】

次に、そのＯＳ１０７のＯＳプログラムコード１３３と実行プログラムコード１３１と実行環境差異吸収プログラムコード１３２をリンクしてできあがったプログラムコード全体をワーキングメモリ２０２に組み込む。また、予めＯＳ１０７のＯＳプログラムコード１３３が搭載されたワーキングメモリ２０２に、後から実行プログラムコード１３１と実行環境差異プログラムコード１３２をリンクしてできあがったプログラムコードを組み込み、動的にリンクしながら実行させるようにしても良い。

【００７８】

ここで、ハードウェア１０６は、実行プログラムを実行できるような機構を備えていれば良いが、本実施形態では、主にプログラムをワーキングメモリに組み込んだシステムＬＳＩなどを想定している。

【００７９】

以下、図９を用いてこのハードウェア１０６の構成の例について詳細に説明する。

【００８０】

ＣＰＵ１２１は、命令を実行し、計算をおこなうハードウェア１０６のメインのコンポーネントであり、図３で説明したような汎用レジスタ群２２２を備えている。汎用レジスタ群２２２に含まれるのは、引数格納用途レジスタ１２４、一時変数格納用途レジスタ１２５と、戻り値格納用途レジスタ１２６、ユーザ使用汎用レジスタ１２７、スタックポインタレジスタ１２８、プログラムカウンタレジスタ１２９などである。

【００８１】

ＲＡＭ１２３は、いつでも書き込みが可能なメモリエリアである。このＲＡＭ１２３に、スタック領域２０５が実装されている。ＲＯＭ１２２は、通常は書きこむことができず、読みだし専用のメモリエリアである。例えば、ＬＳＩの製造時にのみ情報を書きこみ、通常の実行時には、このメモリエリアからは、情報を読むだけである。ＲＯＭ１２２には、定数格納領域１３４が実装される。そして、ＲＯＭ２０３ｂまたはＲＡＭ１２３の一部には、実行プログラム１１０と実行環境差異プログラム１１１とＯＳのプログラムコードが組み込まれ、ＣＰＵ１２１に読み出されて実行される。

【００８２】

デバイス２０１は、各種機能を持つハードウェア１０６のコンポーネントであり、ハードウェア１０６がシステムＬＳＩの場合には、ＬＳＩ上に構成される特殊機能の処理回路である。具体的には、Ｉ／Ｏ（Input/Output），ＡＳＩＣ（Application Specific Integrated Circuits），ＦＰＧＡ（Field Programmable Gate Arrays），ＤＳＰ（Digital Signal Processor）などが考えられる。

【００８３】

Ｉ／Ｏは、例えば、Ａ／Ｄ変換器、Ｄ／Ａ変換器、ＲＳ２３２－Ｃ処理回路、ＳＣＳＩ処理回路である。ＡＳＩＣは、例えば、ＭＰＥＧ　Ｖｉｄｅｏ符号器、ＭＰ３復号器等の専用処理回路である。ＦＰＧＡは、ハードウェア構成を可変にできるＩＣを言う。ＤＳＰは、ディジタル信号処理専用のＩＣである。

【００８４】

これらのコンポーネントは、信号の共通の通り道であるバス２０３により、情報のやり取りをする。

〔Ｉｍｍｅｄｉａｔｅロード命令〕

本発明は、実行環境の差異を吸収するレイヤにおいて、共通エラーコードを返すときに、Ｉｍｍｅｄｉａｔｅロード命令を用いることに特徴がある。そのため、次に、図１０ないし図１２を用いてＩｍｍｅｄｉａｔｅロード命令の形式と機能について説明する。

図１０は、Ｉｍｍｅｄｉａｔｅロード命令の形式を示した図である。

図１１は、Ｉｍｍｅｄｉａｔｅロード命令の他の形式を示した図である。

図１２は、本発明に係る計算機システム構成方法において、共通化エラーコードとＩｍｍｅｄｉａｔｅロード命令の関係を示す図である。

【００８５】

Ｉｍｍｅｄｉａｔｅロード命令１８１ｉは、Ｉｍｍｅｄｉａｔｅ値（即値、定数データ）を、指定されたレジスタに持ってくる命令である。その命令語としての形式は、例えば、図１０に示されるようにＩｍｍｅｄｉａｔｅ設定部１８２と汎用レジスタ選択部１８３とオペコード部１８４で構成される。オペコード部１８４は、命令の種類および機能を表すコードが入る部分である。汎用レジスタ選択部１８３は、ロードするレジスタの番号を指定する。Ｉｍｍｅｄｉａｔｅ設定部１８２は、Ｉｍｍｅｄｉａｔｅ値を保持する部分である。

【００８６】

ＣＰＵ１２１は、オペコード部１８４から、命令の種類および機能がＩｍｍｅｄｉａｔｅロード命令１８１であることを判別し、Ｉｍｍｅｄｉａｔｅ設定部１８２に指定されたＩｍｍｅｄｉａｔｅ値を、汎用レジスタ選択部１８３で指定された汎用レジスタに格納する。

【００８７】

したがって、コンパイルする以前に共通化エラーコード１３６の値と実行環境差異吸収プログラムコード１３２の戻り値をＩｍｍｅｄｉａｔｅ設定部１８２の扱える範囲の数値と定義しておくと、例えば、Ｃコンパイラ１０８は、定数格納領域データロード命令１４３の代わりに、実行環境の差異を吸収するための共通化エラーコード１３６を、ユーザが記述したアプリケーションに返すために、戻り値格納レジスタに設定する命令などについて、その共通化エラーコード１３６

をＩｍｍｅｄｉａｔｅ設定部１８２に保持するＩｍｍｅｄｉａｔｅロード命令１８１を作成してくれる。

【００８８】

本発明では、図１２に示したように、Ｉｍｍｅｄｉａｔｅロード命令１８１のＩｍｍｅｄｉａｔｅ設定部１８２に共通化エラーコード１３６を定義しておき、そのＩｍｍｅｄｉａｔｅ設定部１８２のＭＳＢを０として表せる正の値の範囲に設定する。

【００８９】

また、ＣＰＵの命令セットとして、図１０に示した形式のＩｍｍｅｄｉａｔｅロード命令１８１ｉの他に、図１１に示した形式を持ち、ロードする汎用レジスタを複数設定できるＩｍｍｅｄｉａｔｅロード命令１８１ｊを有することもあるが、このＩｍｍｅｄｉａｔｅロード命令１８１ｊを、共通化エラーコード１３６の設定のために、Ｉｍｍｅｄｉａｔｅロード命令１８１ｉと全く同様に使用することができる。

〔共通化エラーコードの具体例〕

次に、図１３を用いて本発明で設定する共通化エラーコードの具体例について説明する。

図１３は、本発明の計算機システム構成方法に係る共通化エラーコードの具体例を示す図である。

【００９０】

図１３に示した共通化エラーコードは、Ｉｍｍｅｄｉａｔｅロード命令１８１のＩｍｍｅｄｉａｔｅ選択部１８２として、例えば、１バイト長、２バイト長の場合が想定できる。

〔プログラムコードとＩｍｍｅｄｉａｔｅロード命令の関係〕

次に、図１４を用いてプログラムコードとＩｍｍｅｄｉａｔｅロード命令の関係を説明する。

図１４は、プログラムコードにＩｍｍｅｄｉａｔｅロード命令が埋め込まれているときの計算機システムーＡ１０５ａで取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【００９１】

実行プログラムコード１３１は、アプリケーション本体をコンパイルして作られる実行形式のモジュールであり、実行環境差異吸収プログラムコード１３２は、共通関数をコンパイルしてり、必要なモジュールをリンクして得られる実行形式のモジュールである。これらの実行形式のモジュールは、ハードウェア１０６のＲＯＭ１２２またはＲＡＭ１２３上に実装される。

【００９２】

図１４に示された実行プログラムコード１３１および実行環境差異吸収プログラムコード１３２には、図４により説明した定数格納領域データロード命令１４３の代わりに、Ｉｍｍｅｄｉａｔｅロード命令１８１が利用される。そのＩｍｍｅｄｉａｔｅロード命令１８１内部のＩｍｍｅｄｉａｔｅ選択部１８２は、ＭＳＢ１６４が０になっている。そして、ＭＳＢ１６４以外のＩｍｍｅｄｉａｔｅ部２８１には、共通化インターフェース１０２により定められている共通化エラーコード１３６が埋め込まれている。

【００９３】

したがって、この場合に、共通化エラーコードを定数格納領域に保持する必要がなくなる。

〔Ｃソースプログラムの例〕

次に、図１５および図１６を用いて本発明の計算機システム構成方法の開発環境で準備されるＣソースプログラムの例について説明する。

図１５は、本発明の開発環境の共通化関数のＣソースプログラムの例を示した図である。

図１６は、本発明の開発環境のアプリケーションソフトウェアのＣソースプログラムの例を示した図である。

【００９４】

共通化関数１０１は、ユーザにシステムの機能を提供する関数であり、ユーザに対する関数インタフェースとしては、共通化インターフェース１０２を公開する。共通化インターフェース１０２は、実質上、Ｃ言語仕様でいう関数定義のことであり、インターフェース機能名称２４２、引数のデータ型２４３、戻り値の

データ型２４４を定義している。この図１５に示す例では、インターフェース機能名称２４２は、「WPR_Wait_Mail」であり、引数は二つであり、引数のデータ型２４３として、共に「int」型、戻り値のデータ型２４４は、「unsigned int」型である。

【００９５】

共通化関数のＣソースプログラム２４１には、実行環境の差異を吸収するためのコーディングがなされる。例えば、計算機システムーＡ１０５ａの実行開発環境－Ａ１０４ａ上で動作させる場合には、そのＯＳ－Ａ１０７ａのシステムコール２４５ａをコールするコーディングがなされる。また、計算機システムーＢ１０５ｂの実行開発環境－Ｂ１０４ｂ上で動作させる場合には、そのＯＳ－Ｂ１０７ｂのシステムコール２４５ａをコールするコーディングがなされる。

【００９６】

このように共通化関数１０１を用いることにより、ユーザに対して実行環境の差異を隠蔽することが可能であり、ユーザのアプリケーションからすると共通化インタフェース１０２とその関数の機能のみを意識すれば良い。

【００９７】

実行開発環境－Ａ１０４ａ上で動作させる共通化関数１０１の内部の処理としては、図１５に示されるように、例えばシステムコール２４５ａを介して所望の処理をおこない、その結果、ＯＳ－Ａ１０７ａから戻されるシステムコールのＯＳ－Ａ１０７ａ依存エラーコード１３５ｕ，１３５ｖを判別して、共通化インターフェース１０２定義の共通化エラーコード１３６ｕ，１３６ｖ，１３６ｗを共通化関数１０１から戻り値として返すことにする。これまで述べてきたように、共通化エラーコード１３６の数値は、実行開発環境１０４および計算機システム１０５に全く依存しないように、共通化した定義を用いる。

【００９８】

そして、共通化エラーコード１３６の値を、Ｉｍｍｅｄｉａｔｅロード命令１８１におけるＩｍｍｅｄｉａｔｅ設定部１８２のＭＳＢを０として表せる数値範囲であり、かつ、符号無しと定義するので、戻り値のデータ型２４４と共に少なくともＣ言語仕様でデータ型として用いられるunsigned型で宣言する。図１５の

共通化関数１０１では、汎用レジスタ群２２２と同じｂｉｔ長とし、unsigned int型で記述しているが、共通化エラーコード１３６をunsigned short型、 unsigned char型といった汎用レジスタ群２２２のｂｉｔ長よりも短く定義しても構わない。

【００９９】

次に、この共通化関数１０１を呼出すアプリケーションソフトウェアのＣソースプログラムを示すと、図１６のようになる。この例では、アプリケーションプログラム関数２６１「ApplicationTask」が、共通化関数１０１「WPR_Wait_Mail」を呼出している。

【０１００】

このアプリケーションプログラム関数２６１は、内部で共通化関数１０１を呼出すだけであり、実行環境に依存するコーディング入れないため、アプリケーションレベルの開発においては、全く実行環境、ＯＳ、実行される計算機を意識する必要はない。

【０１０１】

図１６に示した例では、アプリケーションプログラム関数２６１「ApplicationTask」が、共通化関数１０１「WPR_Wait_Mail」を呼出し、戻り値をerror_stateという変数２６３に入れて、その後、それが共通化エラーコード１３６ｕ，１３６ｖのどれにあたるを判定し、その処理を分岐させている。戻り値をerror_stateと言う変数の型は、当然、共通化関数１０１「WPR_Wait_Mail」の戻り値の型と同じであり、共通化エラーコード１３６の定義した型であるunsigned int型である。また、共通化エラーコード１３６の型が、他のunsigned short型、unsigned char型等のときには、その宣言した変数２６３に格納する。変数２６３は、アプリケーションプログラム関数２６１でＣ言語仕様のローカル変数宣言またはグローバル変数宣言されているかどうかに従い、ＯＳ別プログラム開発環境２０１によりそれぞれ、局所変数及び一時変数としてスタック領域１３０またはグローバル変数としてスタック領域１３０以外のＲＡＭ領域に実装される。

【０１０２】

【発明の効果】

本発明によれば、計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際に、既存の命令セットの体系を活かしながら、その共通化コードのインプリメントのためのＲＯＭ効率を良くし、むだな命令を挿入することのない計算機システム構成方法を提供することができる。

【図面の簡単な説明】

【図１】

特開平８－６３３６３号公報の「仮想実行環境システム」のシステム概要図である。

【図２】

異種実行環境においてソフトウェア開発をおこなうときの概念図である。

【図３】

計算機システムーＡ１０５ａで取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【図４】

汎用レジスタでゼロ拡張する際の説明図である。

【図５】

ゼロ拡張された汎用レジスタの値を負にする際の説明図である。

【図６】

汎用レジスタで符号拡張する際の説明図である。

【図７】

符号拡張された汎用レジスタの値をもとに戻す際の説明図である。

【図８】

本発明に係る異種実行環境における計算機システム構成方法のソフトウェア開発環境の処理の流れを説明するための図である。

【図９】

本発明に係る異種実行環境における計算機システム構成方法を実行するためのハードウェア構成図である。

【図１０】

Ｉｍｍｅｄｉａｔｅロード命令の形式を示した図である。

【図１１】

Ｉｍｍｅｄｉａｔｅロード命令の他の形式を示した図である。

【図１２】

本発明に係る計算機システム構成方法において、共通化エラーコードとＩｍｍｅｄｉａｔｅロード命令の関係を示す図である。

【図１３】

本発明の計算機システム構成方法に係る共通化エラーコードの具体例を示す図である。

【図１４】

プログラムコードにＩｍｍｅｄｉａｔｅロード命令が埋め込まれているときの計算機システム－Ａ１０５ａで取扱われるデータ、プログラムコードと、レジスタ、メモリなどの関連をあらわす図である。

【図１５】

本発明の開発環境の共通化関数のＣソースプログラムの例を示した図である。

【図１６】

本発明の開発環境のアプリケーションソフトウェアのＣソースプログラムの例を示した図である。

【符号の説明】

１０…アプリケーションソフトウェア

１１…プログラム本体

１２…置換情報記述部

２０…仮想実行環境実現システム

２１…翻訳部（既存ＯＳ用コンパイラ）

２２…実行部（仮想実行環境実現部）

３０…実行可能プログラム

６０…既存オペレーティング・システム

１００…アプリケーション本体

１０１…共通化インターフェース

１０２…共通化関数

１０３…実行環境プログラム

１０４…実行環境

１０５…計算機システム

１０６…ハードウェア

１０７…ＯＳ

１０８…Ｃコンパイラ

１０９…リンカ

１１０…実行プログラム

１１１…実行環境差異吸収プログラム

１２１…ＣＰＵ

１２２…ＲＯＭ

１２３…ＲＡＭ

１２４…引数格納用途レジスタ

１２５…一時変数格納用途レジスタ

１２６…戻り値格納用途レジスタ

１２７…ユーザ使用汎用レジスタ

１２８…スタックポインタレジスタ

１２９…プログラムカウンタレジスタ

１３０…スタック領域

１３１…実行プログラムコード

１３２…実行環境差異吸収プログラムコード

１３３…ＯＳプログラムコード

１３４…定数格納領域

１３５…ＯＳ依存エラーコード

１３６…共通化エラーコード

１３７…実行プログラム開始番地

１３８…実行プログラム終了番地

１３９…ジャンプ命令

１４０…ソフトウェア割り込み命令またはジャンプ命令

１４１…リターン命令

１４２…実行環境差異吸収プログラムコードへの戻り先番地

１４３…定数格納領域データロード命令

１４４…比較命令

１４５…実行プログラムへの戻り先番地

１６１…下位部

１６２…上位部

１６３…２の補数変換処理命令

１６４…ＭＳＢ

１６５…ゼロ埋め込み命令

１８１…Ｉｍｍｅｄｉａｔｅロード命令

１８２…Ｉｍｍｅｄｉａｔｅ設定部

１８３…汎用レジスタ選択部

１８４…オペコード部

１９０…ライブラリ

１９１…ＯＳコンフィグレータ

２０１…デバイス

２０２…ワーキングメモリ

２０３…バス

２１０…ＯＳ別プログラム開発環境

２１１…プリプロセッサ

２１２…Ｃコンパイラ

２１３…アセンブラ

２１４…リンカ

２２２…汎用レジスタ群

２４１…共通化関数のＣソースプログラム

２４２…インターフェース機能名称

２４３…引数のデータ型

２４４…戻り値のデータ型

２４５…システムコール

２６１…アプリケーションプログラム関数

２６２…引数

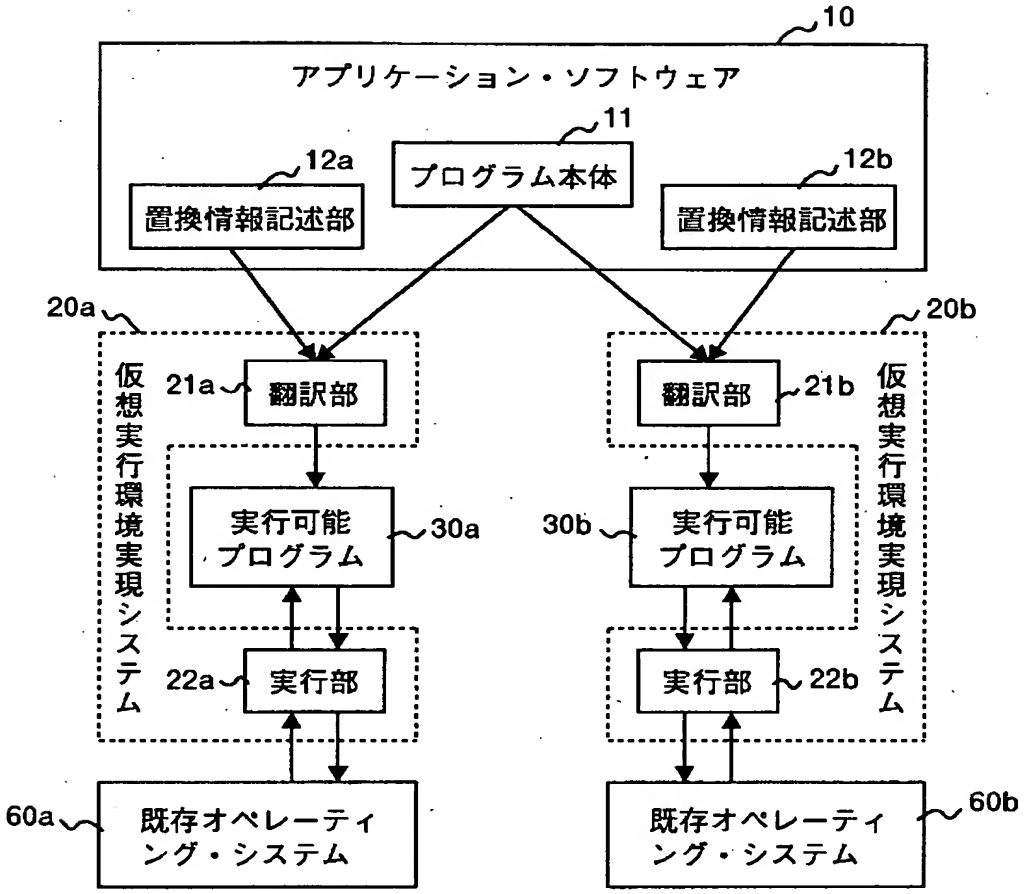
２６３…変数

２８１…ＭＳＢ以外のＩｍｍｅｄｉａｔｅ部

【書類名】図面

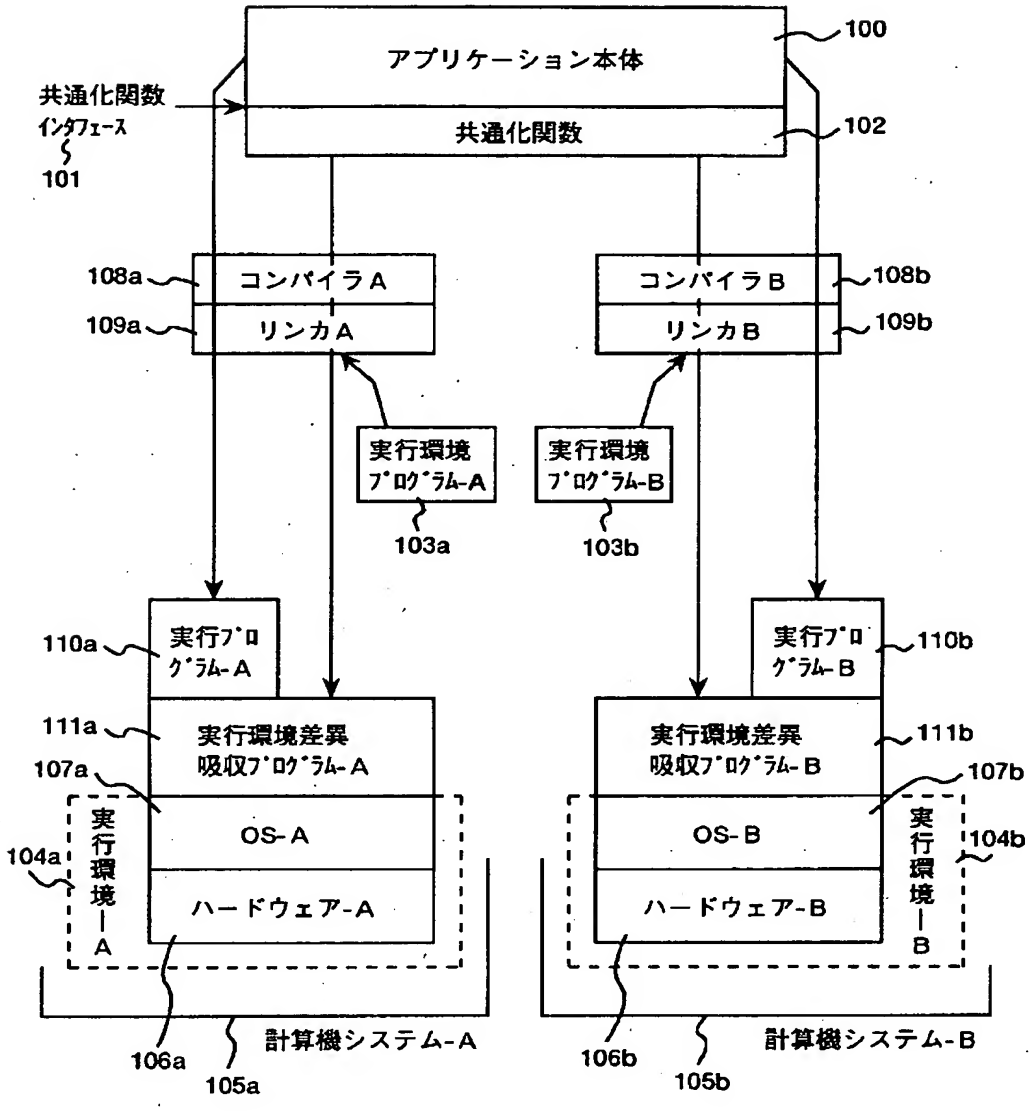
【図１】

図　1


10
アプリケーション・ソフトウェア
11
プログラム本体
12a
置換情報記述部
12b
置換情報記述部
20a
仮想実行環境実現システム
21a
翻訳部
実行可能プログラム
30a
22a
実行部
60a
既存オペレーティング・システム
20b
仮想実行環境実現システム
21b
翻訳部
30b
実行可能プログラム
22b
実行部
60b
既存オペレーティング・システム

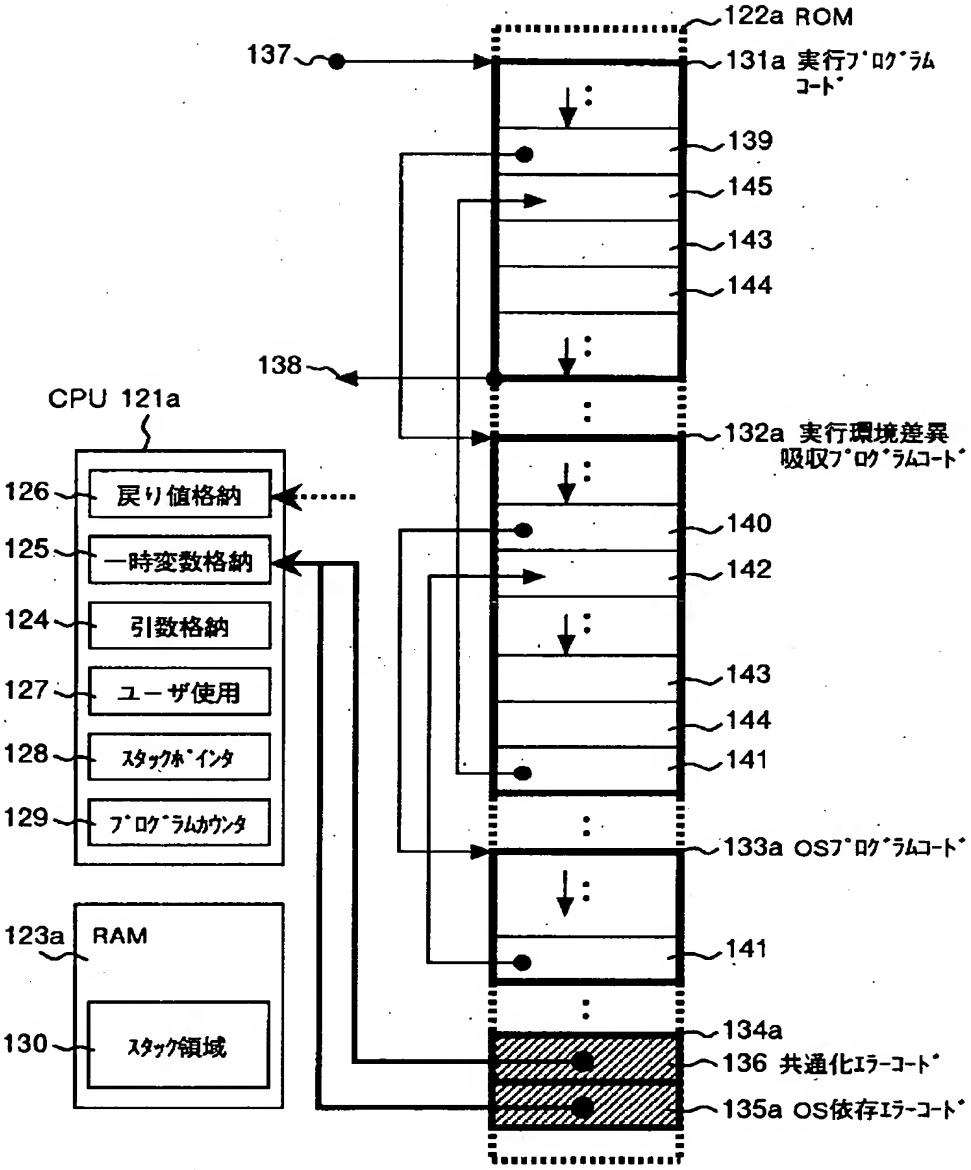
【図２】

図　2


アプリケーション本体
100
共通化関数
インタフェース
101
共通化関数
102
108a
コンパイラA
108b
コンパイラB
109a
リンカA
109b
リンカB
実行環境
ﾌﾟﾛｸﾞﾗﾑ-A
103a
実行環境
ﾌﾟﾛｸﾞﾗﾑ-B
103b
110a
実行ﾌﾟﾛ
ｸﾞﾗﾑ-A
実行ﾌﾟﾛ
ｸﾞﾗﾑ-B
110b
111a
実行環境差異
吸収ﾌﾟﾛｸﾞﾗﾑ-A
111b
実行環境差異
吸収ﾌﾟﾛｸﾞﾗﾑ-B
107a
107b
実行環境－A
OS-A
OS-B
実行環境－B
104a
104b
ハードウェア-A
ハードウェア-B
106a
計算機システム-A
106b
計算機システム-B
105a
105b

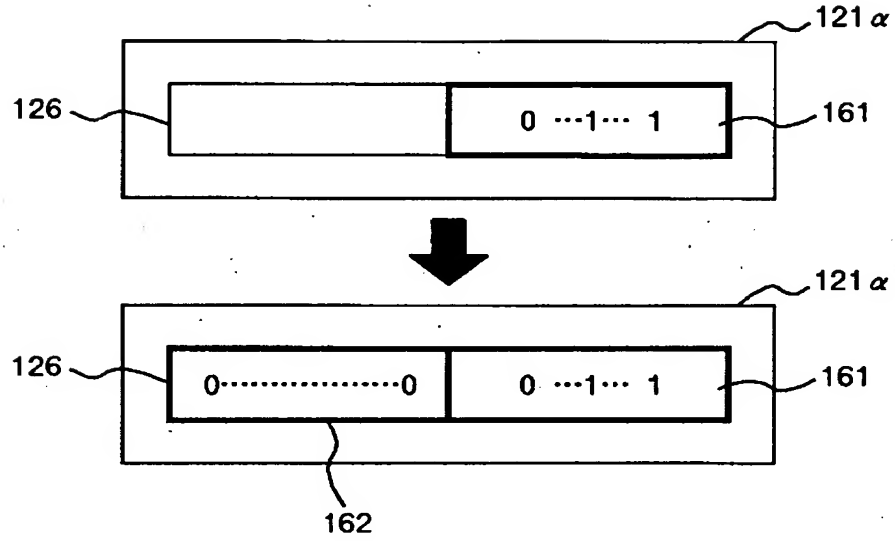
【図３】

図　3


122a ROM
137
131a 実行ﾌﾟﾛｸﾞﾗﾑ
ｺｰﾄﾞ
139
145
143
144
138
CPU 121a
132a 実行環境差異
吸収ﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞ
126
戻り値格納
125
一時変数格納
140
142
124
引数格納
127
ユーザ使用
143
144
128
ｽﾀｯｸﾎﾟｲﾝﾀ
141
129
ﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
133a OSﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞ
123a
RAM
141
134a
130
ｽﾀｯｸ領域
136 共通化ｴﾗｰｺｰﾄﾞ
135a OS依存ｴﾗｰｺｰﾄﾞ

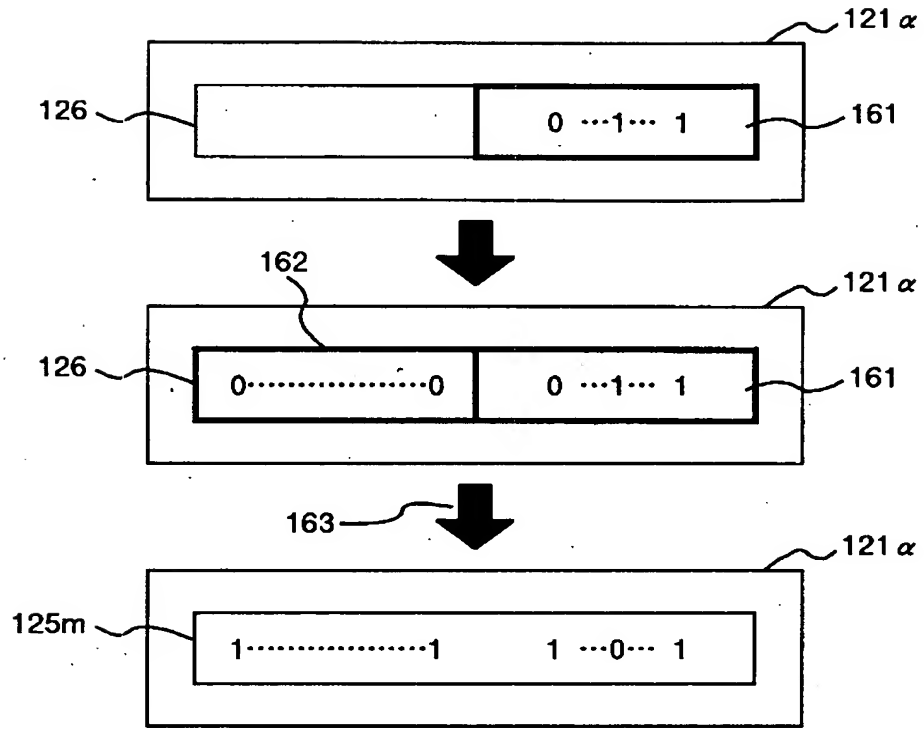
【図４】

図　4



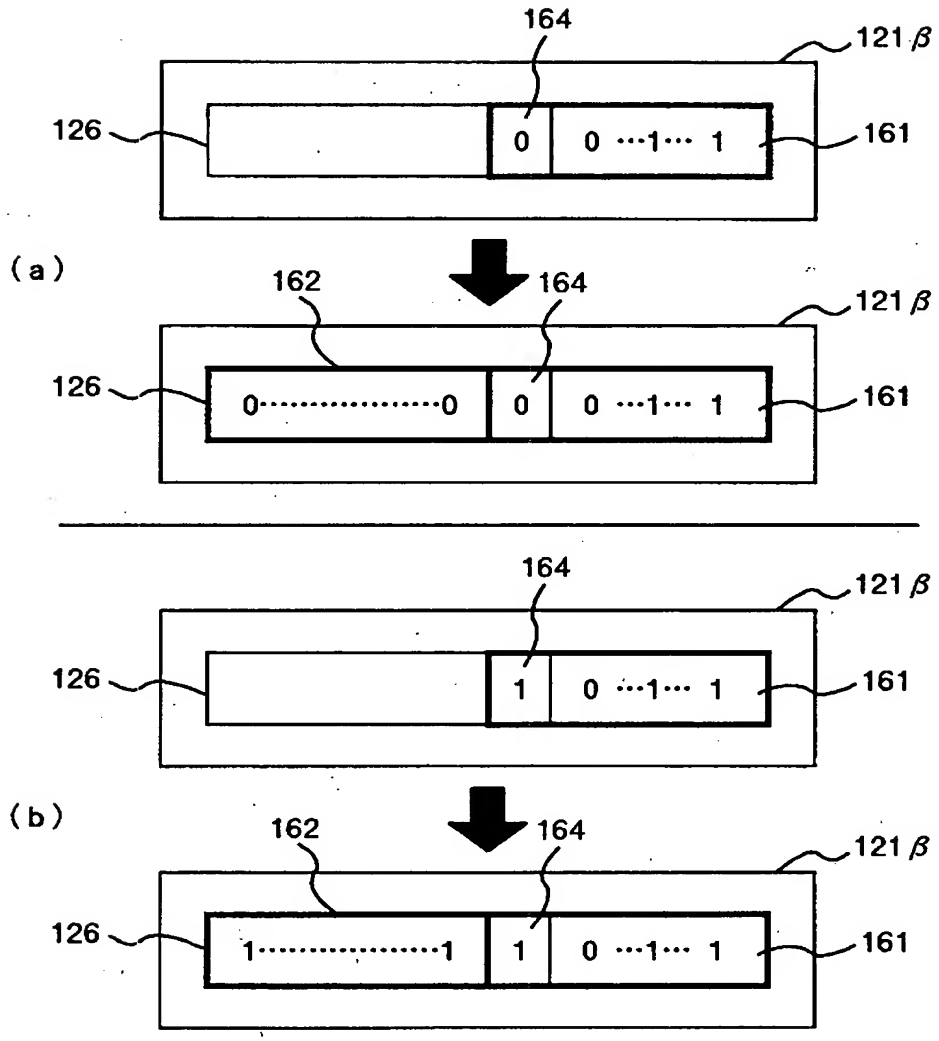
【図５】

図　5



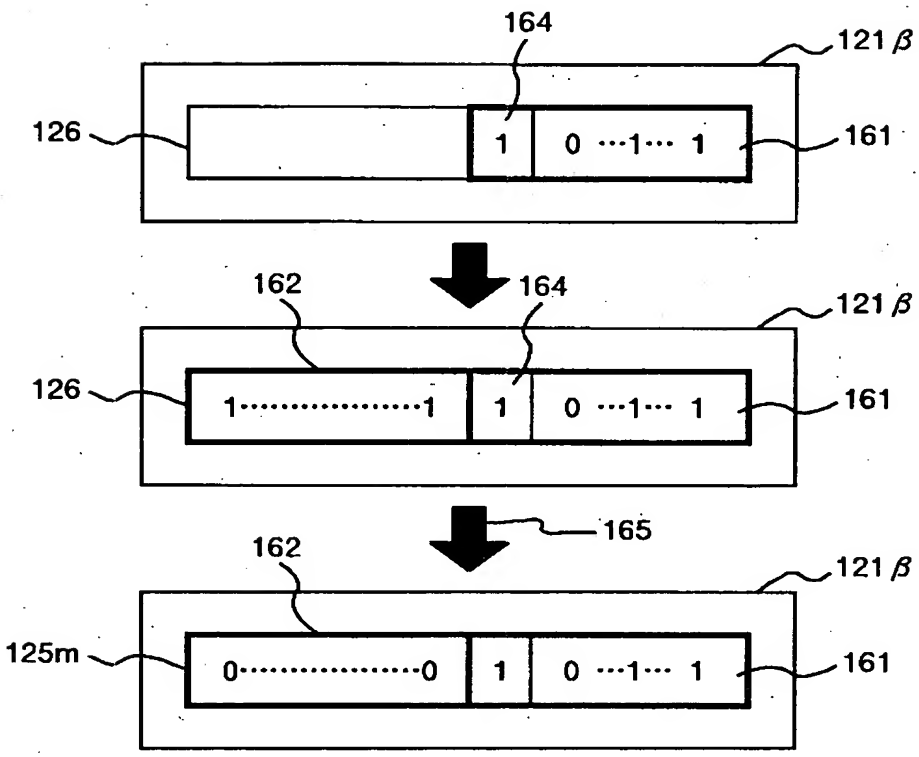
【図６】

図　６


164
121β
126
0
0 ···1··· 1
161
(a)
162
164
121β
126
0·····0
0
0 ···1··· 1
161
164
121β
126
1
0 ···1··· 1
161
(b)
162
164
121β
126
1·····1
1
0 ···1··· 1
161

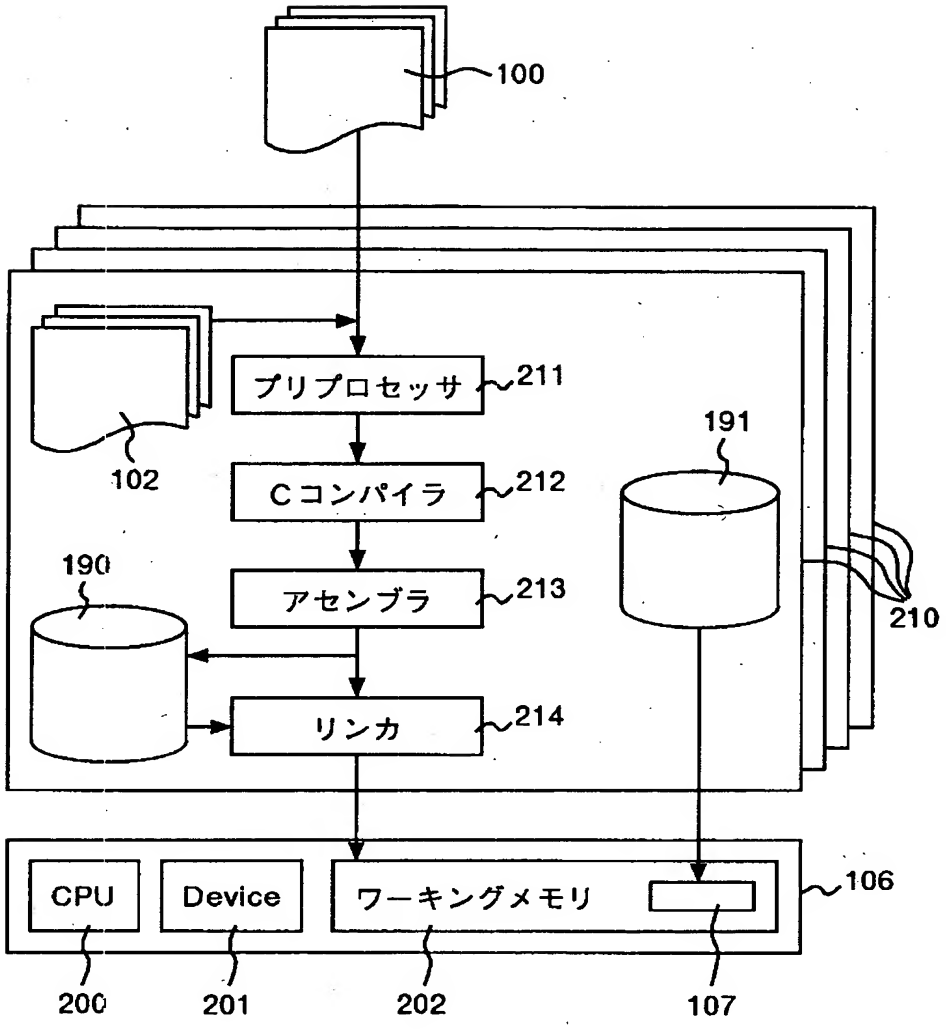
【図７】

図　7


164
121β
126
1
0 ···1··· 1
161
162
164
121β
126
1·····1
1
0 ···1··· 1
161
165
162
121β
125m
0·····0
1
0 ···1··· 1
161

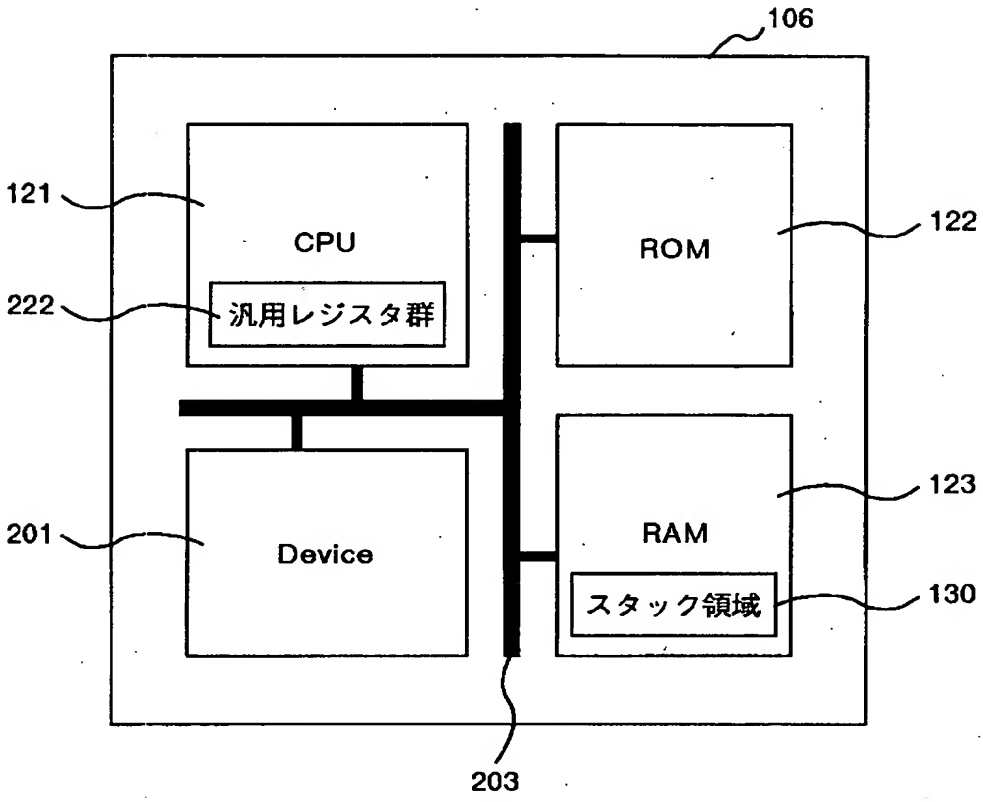
【図８】

図　8


100
プリプロセッサ
211
102
Cコンパイラ
212
191
190
アセンブラ
213
210
リンカ
214
CPU
Device
ワーキングメモリ
106
200
201
202
107

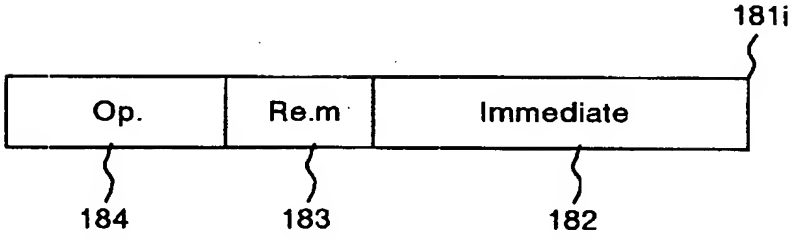
【図９】

図　９


106
121
CPU
222
汎用レジスタ群
ROM
122
123
201
Device
RAM
スタック領域
130
203

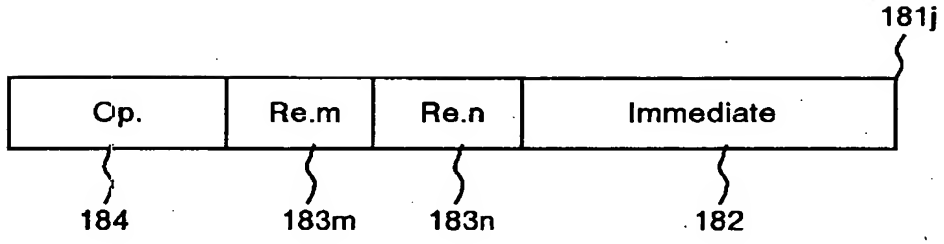
【図１０】

図　１０


181i
Op.
Re.m
Immediate
184
183
182

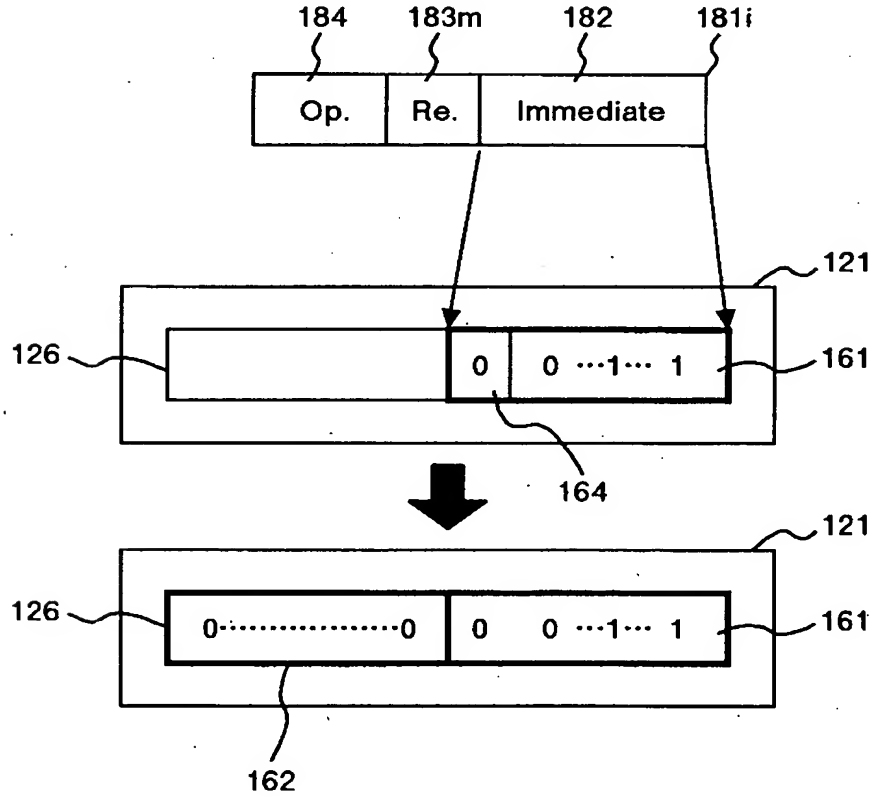
【図１１】

図　１１



【図１２】

図　１２



【図１３】

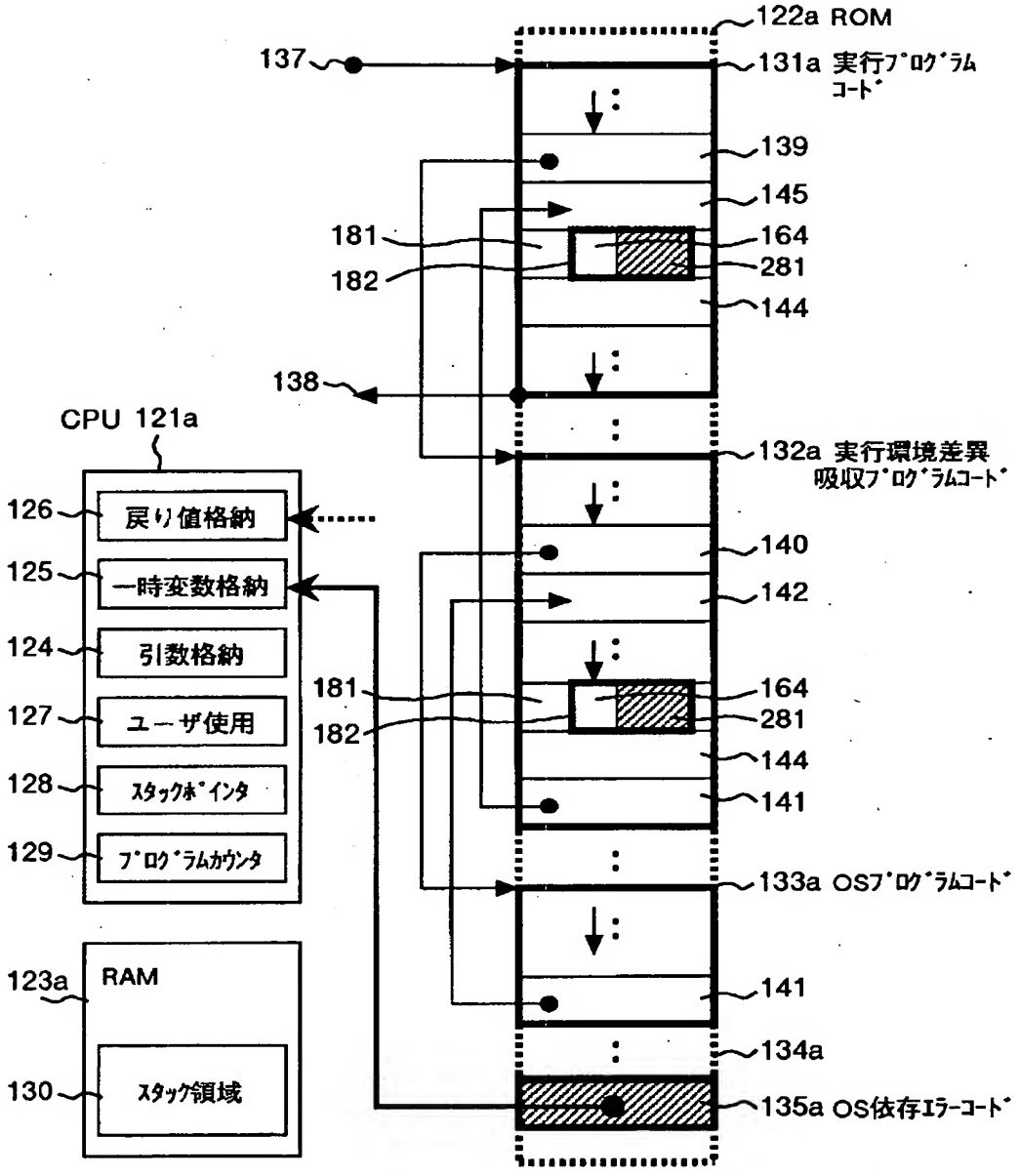
図　１３

エラーコード名称	値	意味
WRP_OK	0X 00	正常
WRP_ER_STACK	0X 01	スタックエラー
WRP_CREATE	0X 03	タスク生成失敗
WRP_ER_MEM	0X 05	メモリエラー
WRP_ER_OTHER	0X 06	その他エラー
WRP_ER_DELETE	0X 0D	タスクが削除されているので、実行不可

0X：16進数を表す

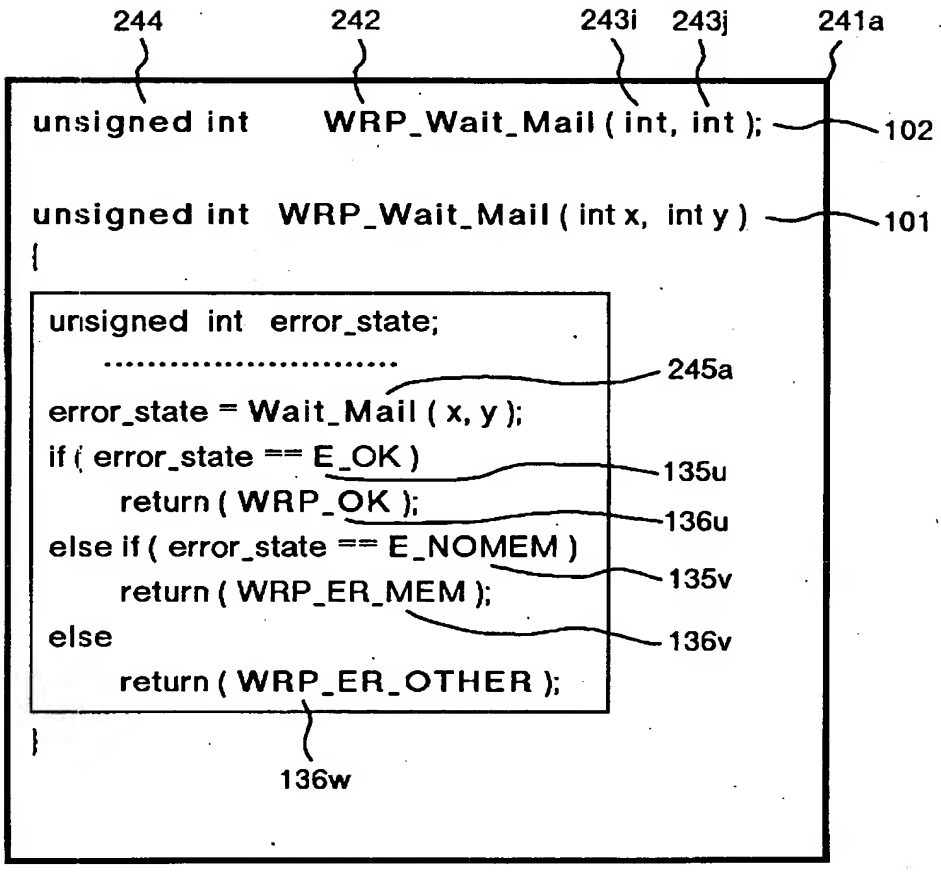
【図１４】

図　１４


122a ROM
137
131a 実行ﾌﾟﾛｸﾞﾗﾑ
ｺｰﾄﾞ
139
145
181
164
182
281
144
138
CPU 121a
132a 実行環境差異
吸収ﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞ
126
戻り値格納
125
一時変数格納
140
142
124
引数格納
181
164
127
ユーザ使用
182
281
144
128
ｽﾀｯｸﾎﾟｲﾝﾀ
141
129
ﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
133a OSﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞ
123a
RAM
141
134a
130
ｽﾀｯｸ領域
135a OS依存ｴﾗｰｺｰﾄﾞ

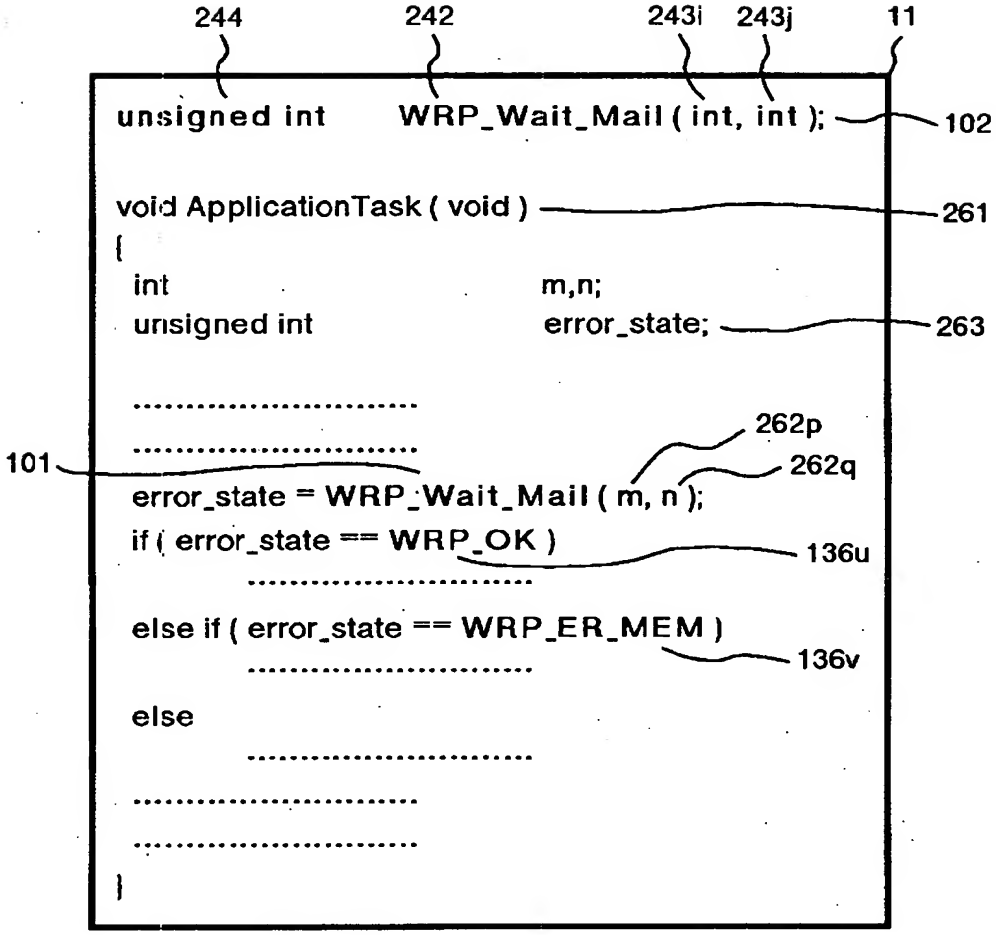
【図１５】

図　１５


244
242
243i
243j
241a
unsigned int WRP_Wait_Mail (int, int);
102
unsigned int WRP_Wait_Mail (int x, int y)
101
{
unsigned int error_state;
.....
245a
error_state = Wait_Mail (x, y);
if (error_state == E_OK)
135u
return (WRP_OK);
136u
else if (error_state == E_NOMEM)
135v
return (WRP_ER_MEM);
136v
else
return (WRP_ER_OTHER);
}
136w

【図１６】

図　１６


244
242
243i
243j
11
unsigned int WRP_Wait_Mail (int, int);
102
void ApplicationTask (void)
261
{
int m,n;
unsigned int error_state;
263
101
262p
262q
error_state = WRP_Wait_Mail (m, n);
if (error_state == WRP_OK)
136u
else if (error_state == WRP_ER_MEM)
136v
else
}

【書類名】　　要約書

【要約】

【課題】計算機システムの異なった実行環境で実行させるときのアプリケーション・プログラムにエラーコードを返す際に、既存の命令セットの体系を活かしながら、その共通化コードのインプリメントのためのＲＯＭ効率を良くし、むだな命令を挿入することのないようにする。

【解決手段】実行環境差異吸収プログラムから、アプリケーション・プログラムに対応する実行プログラムに返すための共通化エラーコードを、ＣＰＵの命令セットと定義されているＩｍｍｅｄｉａｔｅロード命令で設定可能な数値範囲内の値として定めて、共通化エラーコードを、Ｉｍｍｅｄｉａｔｅロード命令の命令コード内に保持するようにする。特に、共通化エラーコードの値をＩｍｍｅｄｉａｔｅロード命令の設定部のＭＳＢを０となる範囲内に収まるようにする。

【選択図】　　図１２

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所